Implementation and Test of Demand Response using Behaviour Descriptions

Kullmann, Daniel; Gehrke, Oliver; Bindner, Henrik W.

Published in:
Proceedings

Link to article, DOI:
10.1109/ISAP.2011.6082246

Publication date:
2011

Implementation and Test of Demand Response using Behaviour Descriptions

Daniel Kullmann, Oliver Gehrke and Henrik Bindner
Risø National Laboratory for Sustainable Energy, Roskilde, Denmark

***Abstract*—The term Smart Grid describes the effort to enable the integration of large numbers of renewable distributed energy resources into the power grid. The fluctuations inherent in renewable energy resources imply the need to also integrate the demand side actively into the control of the power system. For this effort to succeed, a new control infrastructure has to be put into place. The power system is a distributed system, and it needs a sophisticated communication framework to cope with communication problems, such as delays and failures. Recently, behaviour descriptions have been proposed as an alternative to synchronous communication in power systems, especially with small distributed energy resources. This article presents an implementation of behaviour descriptions and an experiment that has been carried out to evaluate the feasibility of such an approach.**

***Index Terms*—Smart grids, Communication systems, Intelligent systems**

I. Introduction

Electrical power systems face the challenge of integrating large numbers of distributed energy resources (DER). The power production of most renewable energy sources such as wind turbines and photovoltaics fluctuates. These fluctuations have to be balanced in some way, which leads to the desire to include the demand side actively into the control of power systems. This means that the behaviour of large numbers of very different components, located mostly at the distribution level, has to be coordinated.

The term *Smart Grid* has been coined to describe a collection of possible technologies to address the challenges for the electrical power system. It has no single authoritative definition, but most descriptions (e.g. [1]) agree on some common elements: Usage of information and communication technology (ICT), control of the consumption side, improvement of grid reliability, enabling market-driven approaches, integration of different kinds of DER units (generation, consumption and storage).

Control implies communication: the distributed components in the system have to be coordinated to achieve the goal of a stable system. Different control concepts put different demands on the underlying communication system: Most importantly, the communication has to be fast enough for the specific control system, both in terms of latency and bandwidth. Conversely, the existing communication infrastructure may also require modifications to the control system: If communication failures are possible, the control system must be able to handle them gracefully.

Recent work [2], [3] has suggested that smart grid applications may require more abstract, higher-level forms of communication than provided by the established standards for power system communication such as IEC 61850 and IEC 61970. This aligns well with other ongoing work: e.g. a recent effort to create a "Smart Inverter" standard which defines a certain set of services that inverters should be able to provide [4], and research to extend the use of the "function blocks" originally defined in IEC 61499 for use in production automation systems, to the automation of power systems [5].

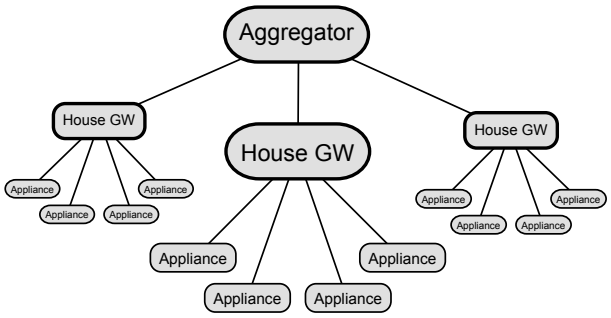

Fig. 1. Assumed System Structure

This article assumes a specific structure of a control system for demand-side components: An aggregation hierarchy that controls many, mostly small, resources. This is exemplified in Figure 1. The components that are controlled, in this case household appliances, are directly controlled by a house gateway. Because the capacity of a single household to provide system services is not large enough to be of interest to the system, an aggregator aggregates the capabilities of multiple households. This aggregator controls the components indirectly, via the house gateway.

II. State of the Art

A number of approaches have been proposed to realise the Smart Grid vision. Some of those use power markets as a central concept, such as the PowerMatcher system [6] or the DEZENT project [7]. Other approaches split up the system into smaller parts that are easier to control; examples for these are Microgrids [8] and the Cell controller [9].

Aggregation approaches have been suggested as a way to cope with the complexity of the system. An example for this are Virtual Power Plants (VPPs), e.g. [10].

All these approaches assume implicitly that communication just happens; the possibility of failure is silently ignored. However, a control structure that can allow failures to happen increases the reliability of a system.

Todays control systems use synchronous communication, where a controlled device responds immediately to the control command of a controller. An example for this is the classical closed-loop control, as employed e.g. in the droop control of a generator. Here, real-time communication between controller and controlled system is important because effects of control impulses have to be visible to the controller (via the feedback loop) as soon as possible.

Work has started recently to create a "Smart Inverter" standard [4] whose idea is to provide a standard set of services that all inverters should provide, among others connect/disconnect, power factor setting and Volt/VAR settings. Standards like these are the foundation on which intelligent, reliable, and distributed control systems can be built. The standard does not describe how to organize communication in the power system to build an intelligent control infrastructure, and it does not address the problem of unreliable communication.

Much research is going on in devising control algorithms for demand-side components, such as heaters and fridges [11] [12] [13]. These control systems must in some way be coordinated to provide a or maximise the benefit for the power system.

[5] proposes to use IEC 61499, the "function block" standard, for defining and providing standardised functionality in power system components. Using the standard is a flexible way to create units ("function blocks") that provide certain functions, but the standard does not contain means to flexibly activate that functionality.

A different approach is to use dynamic power prices to control the behaviour of many units. A dynamic power price contains important information about the state of the power system, in just a single value. The FlexPower project [14] has recently started its work. Its goal is to investigate how a dynamic power price that changes every few minutes can be used to schedule regulating power from small distributed energy resources and from demand side resources.

III. Behaviour Descriptions

As stated above, communication is an essential part of the control of a distributed power system. In order to use the full technical potential of a controllable energy resource, it has to be individually addressable (unicast), and the communication must be bidirectional in order to close the control loop. Depending on the services offered by the resource, a minimum bandwidth and/or maximum communication latency may be required. Generally, the communication must be reliable. If the number of controlled energy resources is large, communication must most importantly be low-cost.

For small resources such as controllable loads at the household level, these requirements are difficult to fulfil at the same time. Typically, a solution will be chosen based on cost, and the control system will be limited by the properties of the communication channel.

A solution to this dilemma can be found using a divide-and-conquer approach: If the individual communication acts, which together form a communication process, could be organised in such a way that each of the individual communication acts has lower requirements to the communication media compared to the whole communication process, a combination of different low-cost technologies could offer equal performance to a single, more expensive technology.

In particular, the communication for many control tasks can be separated into communication acts which require unicast addressing and bidirectional communication but are not time-critical, and other communication acts which demand reliable communication with low latency but can be served by a unidirectional broadcast medium. Both technologies are inexpensive and readily available: consumer DSL connections offer unicast and bidirectional communication but are unreliable and do not give any latency guarantees. Broadcast media have been used for the control of power systems for decades; the most common variants are FM transmitters and tone-frequency ripple control systems. These provide high reliability and defined latency, but do not support bidirectional communication or unicasting.

We will call the first type of communication act a "behaviour description" and the second type a "trigger signal". The trigger signal contains very little information, and is either a locally observable quantity or can be transmitted by unidirectional broadcast medium. The behaviour description, on the other hand, provides a flexible way to specify behaviour. It tells a component how to react to certain situations. An example is to couple power consumption of a heater to a dynamic power price: When the price is high, the consumption is reduced as much as possible; when the price is low, the consumption is increased as much as possible.

The behaviour is negotiated between a component and its supervisory controller. The negotiation happens some time before the behaviour is actually needed. When the behaviour is active, the local component just react to situations as described in the behaviour description. The decisions are based on the trigger signals the component can observe, such as system frequency, voltage, and dynamic power price. The component's ability to take autonomous decisions removes the need for time-critical communication between component and supervisory controller.

The negotiated behaviour depends on the capabilities of the controlled component, i.e. the kinds of services it can provide, and on the constraints the component has to comply with, such as owner preferences or constraints coming from the environment. An example for an owner preference is that the temperature in a house should be in a certain range.

Behaviour descriptions can be interpreted as contracts between a supervisory controller and the components it controls. A contract is a binding document, and so is a behaviour description: Once a component has accepted a behaviour description, it has to adhere to it. This way, the supervisory controller knows how a component will react to certain situations even though there is no continuous communication between the two parties. Behaviour descriptions are also called "policies" due to the similarity to contracts.

Behaviour descriptions offer a good compromise between having autonomous behaviour of the single components, and still being able to control how they behave. This is a crucial characteristic for power systems, because the behaviour of components has to be coordinated to maintain an overall

stable system, while the complexity of the system favours a distributed approach to control.

More complete rationales for using behaviour descriptions have been given in [2] and [3].

The basic building blocks of behaviour descriptions are the services that components can offer to the rest of the system. Some common way to specify and activate services is needed. Components use specifications of services to tell their aggregator what it is they are able to do. Service activations are what the aggregator then sends to the components. This is different from sending the component a command to activate the service; behaviour descriptions contain information about future, not current, behaviour.

Behaviour descriptions offer a number of interesting features. First of all, they enable asynchronous control, i.e. behaviours can be activated without having to communicate directly with the component at that point in time. This makes behaviour-description-based systems much more resilient against communication failures. The second feature is that they enable behaviour on multiple levels. One example for that is a behaviour description containing service activations for a frequency control service and a price-based consumption service. When the frequency is outside its allowed range, the behaviour description will tell the component to react. When the frequency is inside the allowed range, the description allows the component to react to a dynamic power price, thus optimising the electricity costs for the component. Behaviour descriptions can of course contain more than two such levels. Another example for flexibility is letting the component choose from several options for a service. An example for that is to send a list of schedules to the component, and having the component choose a certain schedule, depending on some value that can be obtained by the component (e.g. a broadcast schedule number).

IV. Using Rules to Implement Behaviour Descriptions

There are many options for how behaviour descriptions can be expressed. For this work, a rule-based system [15] has been chosen, where a behaviour description consists of a set of rules. A rule has the form `if` (*condition*) `then` *actions*. The `if` part contains the condition of the rule, i.e. a Boolean expression. When the expression evaluates to true, the actions in the `then` part are executed. The basic structure of a rule set is illustrated in Figure 2. The rule base is embedded in the environment of the component. The conditions reference values from the environment, and the actions change the environment.

Often, several rules apply to the same situation. If the actions of those rules conflict, a single rule has to be selected for execution. This process is called disambiguation. The easiest disambiguation strategy is the "first-one-wins" strategy: The rules are set in a particular order, and they are evaluated in that order. The first rule whose condition is true is the one to be executed.

An important part of the rule-based system is the environment the rule is situated in; without it, a rule would have no access to the state of the physical system (e.g. measurements) and would not be able to perform any actions. Because behaviour descriptions are meant to be a generic expression of control, it is important that the environment is standardised. This implies the need for a common ontology. A suitable environment consists of the following parts:

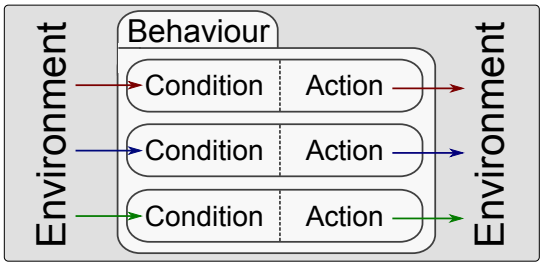

Fig. 2. Behaviour Description with Rules

- A facility to get access to events generated by the outside world, such as local measurements and broadcast messages received.
- Descriptions of services (e.g. "Frequency Response")
- Descriptions of service instantiations, i.e. service descriptions plus a set of parameters needed to use the service (e.g. the droop curve used in a droop controller)
- A mechanism to activate a particular service

Rules are a good way to express behaviour descriptions: First of all, rules are easy to understand for both humans and computers, making human supervision of the control system possible. Rules can also be generated easily with a computer program, because of their strict structure. Flexibility is a key feature of behaviour descriptions, and rules do provide this flexibility: rule sets can be arbitrarily complex, and rules can be adapted to obey local constraints of components.

V. Policy Framework

The implementation of the example case explained in the next section is based on the policy framework, a Java library developed by the authors. It provides a mechanism for using arbitrary behaviour descriptions in a control system. The policy framework was originally developed on top of the JADE multi-agent platform, but has since been extended to also support a simple http-based transport layer which is easier to configure and use. The framework does not put any constraints on the type of behaviour descriptions that can be used; this allows for experimentation with different ways of expressing them.

The policy framework implements the protocol for negotiating policies. The protocol consists of a number of steps that components have to go through:

1) The component is new, and registers at its aggregator
2) The aggregator starts a negotiation round
3) The component sends information about itself to the aggregator: the state it is in, the services it supports and the constraints it has to comply with
4) The aggregator creates a behaviour description/policy and sends it to the component
5) The component accepts the behaviour description

6) The component activates the behaviour description as soon as it becomes valid
7) Before the validity interval expires, the component starts a new negotiation

To implement a specific control system, several classes have to be implemented that have to be registered in the framework during runtime:

1) The actual behaviour description(s) used; this includes serialisation of the descriptions into a text format that is transferred between server and client
2) Descriptions of the components of the client side; this includes the description of supported services, and of the constraints the clients have to obey
3) The server implementation; this part is responsible for creating the behaviour descriptions for each client
4) The client implementation; it provides the component descriptions from item #2, receives the behaviour descriptions, and acts according to them

VI. Implementation

A proof-of-concept demonstration for the control of heaters in a household has been implemented at the SYSLAB facility [16]. SYSLAB is a small power system designed to facilitate the development and testing of distributed control algorithms for power systems. A part of SYSLAB is PowerFlexhouse, an intelligent office building equipped with a large number of sensors and various types of controllable demand. An embedded controller in the building is able to execute behaviour descriptions. In the following example, we will demonstrate how the policy framework can be used to implement reliable dynamic demand response.

In the example, the building is set up to respond to two types of external events: Changes in system frequency, and changes in a dynamic power price. Depending on these parameters and the behaviour description in effect, the building can modify the setpoints of the heater thermostats in individual rooms, in order to decrease or increase the overall power consumption. The system frequency is measured directly at the building's grid connection point. The Nordic power system does not currently have a mechanism for the generation or broadcast of fast dynamic power price signals, although such a mechanism is likely to exist in the future. For the sake of this demonstration, a real-time artificial price signal has been generated, based on production and consumption data in the Danish power system. The price calculation is based on the rationale that a high penetration of wind energy works towards lower spot prices, since the marginal production cost of wind energy is very low.

The implementation uses rules that access service objects as black boxes. That means that the actual implementation of the service is unknown to the rule, and only accessible through a uniform interface. In this way, any kind of service can be used in a rule-based system, without having to expose the implementation details of the service.

The syntax of the rules comes from the Drools expert system, a Java-based rule engine [17]. The Drools system supports two different syntaxes, one a XML format, the other one more programming-language like, which is the one used here.

```
package heatercontrol;

rule "setup"
  when
    registrar: Registrar() and
    ctrl : HeaterController() and
    source1 : SystemFrequency() and
    source2 : PowerPrice()
  then
    registrar.register(
      new FreqControl(
        new Thresholds(49.98, 50.01),
        source1, ctrl) );
    registrar.register(
      new PriceCtrl(
        new Thresholds(55.745, 64.546),
        source2, ctrl) );
end

rule "frequency-1"
  activation-group "hc"
  when
    ctrl : FreqControl ( activate == true ) and
    sched : HeaterController()
  then
    sched.use( ctrl );
end

rule "price-2"
  activation-group "hc"
  when
    ctrl : PriceCtrl( activate == true ) and
    sched : HeaterController()
  then
    sched.use( ctrl );
end
```

Fig. 3. Simplified Example Rule Base

Another issue is how to provide the rule bases with the environment they have to work in. In our example case, the environment consists of the following parts (compare this with the list of environment parts in section IV):

- Measurements of frequency and dynamic power price
- Abstractions of the frequency control and price reaction services
- A mechanism to tell the system to activate a certain service, i.e. a service scheduler

The two kinds of services (frequency control and price control) have a similar API for usage in the rule bases:

- A method to create an instance of a service
- Methods to start and stop the service; these are used by the service scheduler
- A method to evaluate whether a service would like to be activated; only the first such service that is found in the rule base is actually activated

Since the implementation is based on Java, the environment of the rules is also specified using Java interfaces and classes. The measurements of frequency and power are provided by a unified interface which can inject measured (or received) values into the services. The services provide a common interface which has three main functions: a method to inject measured values into the service, a method to ask a service whether it wants to be activated, and a means to activate a service. The service activation is provided by a scheduling

agent that is always present in the environment of the rule base.

A big implementation issue is how to get service instances with the right parameters into the rule base. The services need certain parameters to be able to function; in the example case, both services worked with low/high thresholds, so two parameters are needed for each service instantiation. The problem is how these service instances are put into the environment of the rule base. It would have been possible to send additional information about the services along with the behaviour description, but this meant that a separate data format for service activations would be necessary. The option that is implemented adds an additional "setup" rule to the rule base that is triggered exactly once. This extra rule contains code to initialise the services and to register them in the environment.

An example rule base is shown in Figure 3. This is a cleaned-up version of a rule base that was actually used in the experiment; names and values have been shortened to make it more readable. The first rule `setup` is the setup rule mentioned in the last paragraph; it sets up the two service activations with the appropriate thresholds and registers them in the environment. The other two rules, `frequency-1` and `power-1`, are the actual work rules of the rule base. They just evaluate whether the service in question wants to be activated (i.e. the measurement is outside the service deadband), and if this is true the service scheduler is instructed to activate the service. The two rules exclude themselves mutually (this is done with the `activation-group` directive), so only the first of them that fires is activated. The rule base is very simple because the work of the services is hidden inside the blackbox of the service activation implementation.

VII. Case Experiment

The experimental setup of the example case controlled the eight rooms in PowerFlexhouse individually using behaviour descriptions. Each room was controlled by its own client which offered maximally two services to the system: a frequency control service, and a service that reacted to the dynamic power price. The price reaction service was offered by all clients, but not all offered the frequency control service. This was done to demonstrate that behaviour descriptions can in fact be adapted to the capabilities of the individual components. Both offered services are parametrised by high and low thresholds. The services are kept simple on purpose, because it was not the purpose of the experiment to demonstrate complex control algorithms, but rather to show that a system based on behaviour descriptions actually works, and to evaluate how behaviour descriptions can combine simple services to allow for more complex behaviour.

A server program running on a different computer calculated the behaviours for the single rooms. The thresholds for the services are calculated in a way to provide smooth reaction to changes in frequency and power price.

The policies for the frequency service change every two hours: Three different widths of the frequency controller deadband are used. This was done to demonstrate the flexibiulity of policies. This can be seen in the middle plot of figure 5.

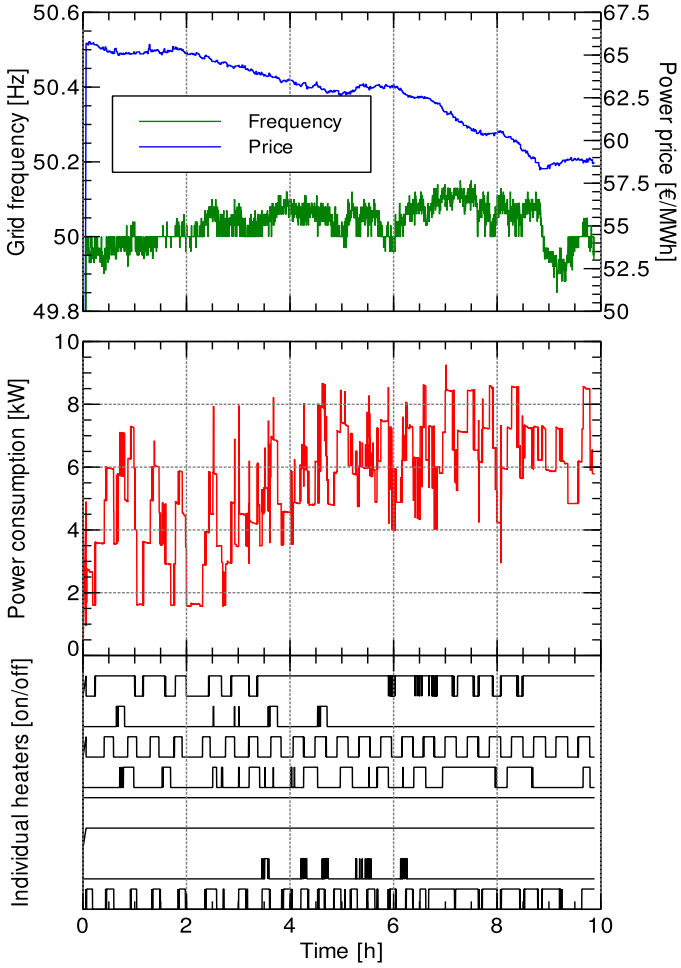

Fig. 4. Overall results

VIII. Observations

The plots in figure 4 contain measured data from a single, 10-hour run of the experiment. During the experiment period, the power price input signal decreases steadily with few and small upward excursions. The system frequency input shows a slowly increasing trend, followed by a marked drop towards the end of the period.

In the second subfigure, the overall power consumption of the building exhibits a strong oscillatory pattern caused by the underlying thermostatic control of the heaters. A correlation between power consumption and system frequency can be observed. The price, on the other hand, does not appear to have a strong impact on the consumption. This can be explained by the fact that the price controller only acts as a secondary control loop, and is overridden whenever the primary frequency controller is active. During the period of the experiment, this was the case for most of the time.

The switching state of the individual heaters is shown in the bottom subplot. Due to low outside temperatures during the experiment, some heaters can be seen to be on almost all of the time.

Figure 5 details the behaviour of a single room during the same experiment. In the first hours of the experiment, the temperature setpoint (red) remains largely at 19°C due to

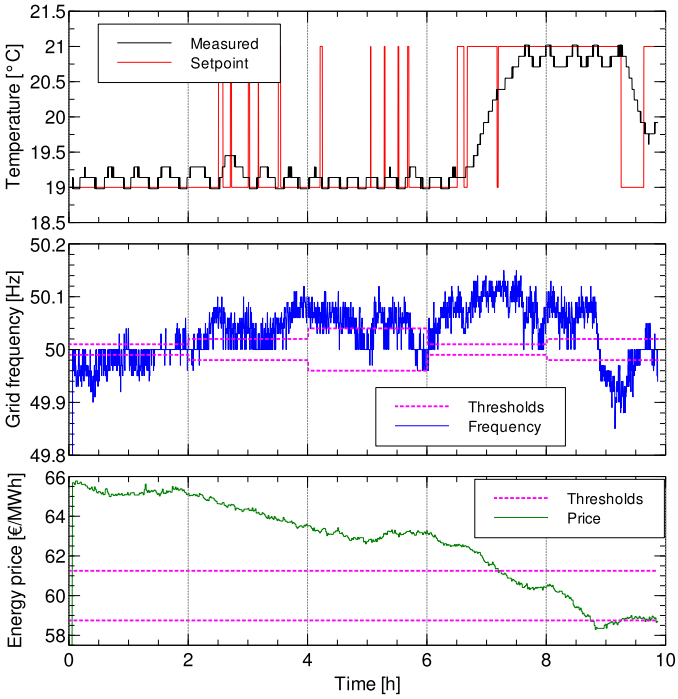

Fig. 5. Results for Room #6

the high power price and low system frequency. The room temperature (black) follows the setpoint within the hysteresis of the thermostatic control.

The second and third subplots show the relationship between the two input signals - price and frequency - and the thresholds in effect according to the active policy. The two-hour period for policy changes is easy to observe.

In hour 6 of the experiment, an increase in system frequency triggers the upper frequency threshold. At nearly the same time, the power price begins to drop, first below the upper, then below the lower threshold. This leads to an almost constant temperature setpoint of 21°C for the remainder of the experiment.

The observations demonstrate that individual clients do react to the frequency and power policies sent to them through the policy framework. However, more and longer experiments as well as a deeper analysis of measured data is needed to determine and quantify the precise effect of changing policies. In the experiment presented in this paper, the thermostatic control of the heaters and external influences such as the outside temperature interfere with the effects caused by the controller. Further statistical analysis is needed to separate these effects.

IX. Conclusion

This article explained the concept behind behaviour descriptions, demonstrated an implementation in the Java programming language, and showed the results of an experimental test of behaviour descriptions in Risø's SYSLAB facility.

Behaviour descriptions enable control systems to deal with communication failures, and they reduce computational and communication load on supervisory controllers, because they decouple supervisory controller and controlled components and make more autonomous, yet still controlled, behaviour of components possible.

Behaviour descriptions don't rely on specific services that components offer to the power system. This is an intentional feature of behaviour descriptions: They provide a container for the flexible and reliable execution of services; the only things needed from the services is the means to describe and to control them.

The experimental results show that behaviour descriptions can have a positive effect on the power system by controlling the power consumption of demand-side consumers. The services used in the experiment are very simple, so the effect of more sophisticated services can be expected to be better.

References

[1] The Modern Grid Initiative, "The Modern Grid Initiative Version 2.0, Conducted by the National Energy Technology Laboratory for the U.S. Department of Energy Office of Electricity Delivery and Energy Reliability," Jan. 2007. [Online]. Available: http://www.netl.doe.gov/moderngrid/resources.html

[2] Daniel Kullmann, Henrik W. Bindner, and Oliver Gehrke, "Towards flexible control and communication of minigrids," in *5th European Conference on PV-Hybrid and Mini-Grid, Tarragona, Spain*, 2010.

[3] Daniel Kullmann and Henrik W. Bindner, "Using Rules in High-level Communication for the Control of Power Systems," in *2nd International Conference in Microgeneration and Related Technologies in Buildings, Glasgow, United Kingdom*, 2011.

[4] Electric Power Research Institute (EPRI), "Standard Language Protocols for Photovoltaics and Storage Grid Integration – Developing a Common Method for Communicating with Inverter-Based Systems," 2010.

[5] N. Higgins, V. Vyatkin, N. C. Nair, and K. Schwarz, "Distributed power system automation with iec 61850, iec 61499, and intelligent control," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 41, no. 1, pp. 81 –92, 2011.

[6] J. Kok, C. Warmer, I. Kamphuis, P. Mellstrand, and R. Gustavsson, "Distributed control in the electricity infrastructure," Presented at International Conference on Future Power Systems 2005, 2006.

[7] H. F. Wedde, S. Lehnhoff, C. Rehtanz, and O. Krause, "Bottom-up self-organization of unpredictable demand and supply under decentralized power management," in *SASO*, S. A. Brueckner, P. Robertson, and U. Bellur, Eds. IEEE Computer Society, 2008, pp. 74–83.

[8] Microgrids & More Microgrids, http://www.microgrids.eu.

[9] energinet.dk, "The Cell Project home page," http://www.energinet.dk/en/menu/R+and+D/The+Cell+Project/.

[10] Danny Pudjianto, Charlotte Ramsay, and Goran Strbac, "The FENIX vision: The Virtual Power Plant and system integration of distributed energy resources," Imperial College, Tech. Rep., 21 Dec. 2006.

[11] Yi Zong, Daniel Kullmann, Anders Thavlov, Oliver Gehrke, and Henrik W. Bindner, "Model Predictive Control Strategy for a Load Management Research Facility in the Distributed Power System with High Wind Penetration -Towards a Danish Power System with 50% Wind Penetration," 2011.

[12] Preben Nyeng, Jacob Østergaard, Mikael Togeby, and János Hethey, "A pooling-based load shift strategy for household appliances," in *24th International Conference on Informatics for Environmental Protection*, 2010, pp. 734–743.

[13] Ontje Lünsdorf and Michael Sonnenschein, "A pooling-based load shift strategy for household appliances," in *24th International Conference on Informatics for Environmental Protection*, 2010, pp. 734–743.

[14] FlexPower project partners, "FlexPower Project Description," http://www.ea-energianalyse.dk/reports/1027_flexpower_project_description.pdf, Tech. Rep., 2010.

[15] J. C. Giarratano and G. D. Riley, *Expert Systems: Principles and Programming*, 4th ed. Course Technology, 2004.

[16] O. Gehrke and H. Bindner, "Building a test platform for agents in power system control: Experience from SYSLAB," in *International Conference on Intelligent Systems Applications to Power Systems 2007*, 2007, pp. 1–5.

[17] The JBoss Project, "Drools," http://www.jboss.org/drools/.